Global Information Assurance Certification Paper

Some Measure of Confidence: Establishing a Patch Testing Infrastructure

GSEC Assignment version 1.4b, Option 2

Greg Hanna

March 2003

1.0 Abstract

This paper will document the establishment of a patch-testing infrastructure at a medium-sized company. The reasons for, the obstacles to, and the steps involved in, establishing the infrastructure will be discussed in detail. The paper will also show the significant and immediate ROI generated by the project, in terms of increased communication, visibility and information assurance, as well as in strictly monetary savings.

2.0 Before Snapshot

2.1 Introduction to the Environment

The organization that I work for is a medium-sized corporation of between three thousand and thirty-five hundred employees worldwide. Our business is software development. The company was founded, grew profitable, and continues to thrive, on the creativity of its product development team (Engineering). Not surprisingly, this has created a strong customer-focus in the information technology department (IT), as well as in the other departments that support Engineering. Consequently, Engineering is accustomed to an environment that puts their needs before the needs of the company as a whole. Therefore, they are, understandably, often reluctant to embrace any initiatives introduced that don't show an immediate personal benefit, or those that inconvenience or encumber them. One such initiative is IT's anti-virus (AV) effort.

Through testing by the Engineering department, it has been discovered that anti-virus software measurably slows system performance, especially during processing-intensive tasks such as software compiling, and patching the OS takes time. Because of this, AV software and OS patches are not universally used within the company, which has created an environment in which there is a constant level of malware lurking on the network waiting for an opportunity. Small outbreaks are common. This situation has created the necessity of two IT teams to combat the proliferation of virii within the company. The first is the Virus Response Team (VRT), a group that keeps abreast of virus alerts, and responds to large-scale outbreaks. The second is the Virus Protection Team (VPT), which works to make recommendations regarding policy, education, and technology to fight the threat of virus infection in a proactive way.

2.2 The VPT and its Mandate

The VPT is a cross-functional "virtual" team comprised of people from every different component of IT, from information security professionals and server administrators to non-technical staff, such as software licensing administrators, project managers, and business analysts. Team membership is on a rotating six-month basis, which encourages timely and tangible results without taxing the membership too heavily for an extended period of time. The patch testing

infrastructure (patch-lab) project is an initiative that comes from this second team, the VPT.

At the beginning of our term, the VPT decided that our focus would be departmental Windows 2000 servers- that is, machines running Windows 2000 Server or Advanced Server that are on the company network, but are owned and managed by Engineering departments rather than IT. During and after attacks by Code Red and NIMDA, it was discovered that there are a large number (almost 4% of our total internal node deployment) of these departmental servers that are vulnerable to such attacks because they are not running anti-virus software, and\or the latest security patches. These servers could be such things as lab machines for testing purposes, departmental file and code servers, or any number of other machine types, right down to laptops used for demonstrating product. Whatever their function, forensic work done after both Code Red and NIMDA showed that they were largely responsible for the spread of both worms, and the resultant network outages. Further, Windows 2000 is the most commonly used OS on the company network, and virtually all of the virus\worm outbreaks recorded on the network in the last two years have been Windows-based. We concluded, therefore, that any blanket solutions for Windows 2000 Server would address the greatest number of vulnerabilities for the resources expended.

The first step was to discover why the servers weren't being properly configured to keep them safe from malware. The VPT conducted interviews with about forty departmental server administrators that were known to members through common projects, or simply by acquaintance. From those interviews, we were able to determine the main issues preventing the administrators from making sure that their machines are adequately protected. We uncovered some fascinating information, and found that we had to break into a number of sub-teams in order to address the most common issues. I headed one of these sub-teams, a three-person group that dealt with services that could be offered to help Engineering administrators to be more effective at keeping critical patches up to date.

Through reviewing the interviews, we identified three main problems. The first was that there was no disaster recovery available to these server administrators, because their servers were not being backed up. The second was that, in a time when seven new patches have already been released by Microsoft less than three months into 2003[1], keeping systems up-to-date was seen as almost a full-time job in itself. The third problem, my problem, was that the server administrators, by and large, were worried that patching systems on the fly from Windows Update had a reasonable chance of causing a system crash from which they had no way to recover (see problem one). As one respondent succinctly said, the administrators needed "some measure of confidence that patching (our systems) won't produce a blue screen upon reboot".

[1] Microsoft KB article 815021, published March 17th, 2003. Describes MS03-007 http://support.microsoft.com/default.aspx?scid=kb;[LN];815021

3.0 During Snapshot

3.1 Defining the Problem

So, how to instill confidence? Unfortunately, it was not a purely technical problem that could be solved by installing a "whiz-bang" tool. After all, if Engineering wasn't satisfied by the fact that Microsoft tests the critical updates published to Windows Update quite extensively, there must be more to the issue than just the technical side.

After discussing the possibilities with my teammates and a few of the Engineering administrators, I came to the conclusion that there were three key elements missing from the current process (or lack thereof) that could help the administrators to be more comfortable applying the patches. The elements were:

a) A level of trust that the patches were tested properly. Whether justified or not, most people in the Engineering community don't trust Microsoft to perform "due diligence" on such things as patch testing.
b) Responsibility in the event that something goes wrong. If a patch were to cause a catastrophic system crash, the administrators need to know that someone will take responsibility, or at least commit to helping them recover their systems.
c) Insight into the testing process. Without knowing what the testing process is for each patch, there is no way for the administrators to know if that process addresses their particular scenarios. Also, the notion of "testing" itself somewhat abstract without a published process. That is, there is a question of what "testing" refers to without a published process.

3.2 Defining the Solution

It quickly became obvious to me that, to provide the missing elements listed above, some verification of patches (beyond Microsoft's publishing them to Windows Update) would be necessary. This meant having a third party do the testing and verification, or doing it ourselves. My runaway first choice was to have someone else do it, by finding a vendor whose product was some sort of testing\verification process. My vision was of a sort of "Seal of Approval" that the vendor would provide on a subscription basis, perhaps published to a website that we could link to. Unfortunately, I found that nothing of the sort was available. I searched the buyer's guide issue of Information Security Magazine, quizzed colleagues, and performed many google searches over a three-day period, all to no avail. While there were a plethora of patch "management" tools available, from Patchlink, to Shavlik (HFNetCHKPro), to Microsoft's own MSUS, there simply wasn't a product or service that was aimed at verifying and\or certifying the quality of the patches themselves. So, with no obvious option, I began to investigate the creation of an in-house patch-testing infrastructure.

3.2.1 Assessing the Current State of Affairs

As a sanity check, I discussed my findings and my proposed solution with the other members of the sub-team, who agreed with my proposal and the reasoning that had led me to it. In order to avoid duplication of effort, and in hopes of making my task easier, I wanted to find out if anything like what I was proposing was already in use, or even in development. One of the members of the sub-team knew that there was some testing of patches being carried out jointly by two separate groups in IT, and he went to speak with them to determine exactly what the situation was. He found that the two groups, Internet Infrastructure and Operations Engineering (the groups responsible for the IT servers on the DMZ and the IT servers on the internal network, respectively) were doing what he called "informal" patch testing. Essentially, the testing was being done on a best-effort basis by one IT engineer from each group. They didn't have the benefit of dedicated testing machines and, due to the volume of work that each was responsible for, they weren't able to test every patch that was released. In addition, the informal nature of the testing meant that it was always considered as a lower priority than other operational tasks, or project-related work that would come up from time-to-time. Further, since there was no established process of communication between the engineers, or out to the rest of IT, the results (albeit, the very existence) of the testing were not visible to anyone but the tester. This meant that not only was the value of the testing diminished by not being widely available, but sometimes the testing of the same patch would be done independently by each IT engineer, duplicating their efforts to no actual gain.

3.2.2 Building on a Solid (but Obscured) Foundation

After the initial discussions with the IT engineers doing the testing, it was immediately apparent that we could build upon, and improve, what was already in place, to the benefit of both IT *and* Engineering. I proposed to the other members of the sub-team that the ideal solution would include the following:

a) A dedicated and uniform testing environment, mimicking the production environment as closely as possible.
b) An established and documented testing process that would address the most possible scenarios with the least possible effort and cost.
c) A process of communicating the results to the entire company community, so that all server administrators, identified and not-as-yet identified, could find and use them.
d) The testing and communication processes needed to be managed going forward, preferably by the two IT engineers who were already involved.

One of the sub-team members wisely pointed out that the last point would require buy-in from not only the IT engineers themselves, but from their respective managers as well. So, to ensure their buy-in, we also had to design our solution so that it would be more comprehensive than what was already established, but have no additional costs to either of the IT groups involved, in staff cycles or dollars.

It was agreed that the VPT sub-team (primarily myself) would be responsible for designing and building the infrastructure in its entirety, including a script to define exactly what and how to do the testing, a database to record the results of the testing and a communications pipeline through which to share the results and any related information with the user community. The VPT structure is temporary, as noted earlier, so I took responsibility personally for any required upgrades or repairs to the lab infrastructure going forward. The IT engineering groups took responsibility for the actual testing of the patches, as well as populating the database with the test results. Approval of the plan was then obtained from the managers of the IT engineers who would be doing the testing, and feedback on the value of the plan was obtained from several key members of Engineering staff. With the high-level plan approved and validated, we set out to put it into place.

3.2.3 VLAN or VMware

One of the obvious requirements of a patch-testing infrastructure is a physical lab in which to conduct the testing. In order to determine exactly what kind of machinery would be required, I decided to first determine exactly what testing would be done. After conferring again with the IT engineers and gleaning from the interview feedback provided by the Engineering administrators, I established the basic testing requirements as follows:
After the patch\hot-fix\service pack to be tested is applied as prescribed, the basic functionality of the following should be tested:

a) Domain-level authentication
b) SQL application authentication
c) DHCP service
d) WINS service
e) DNS service
f) IIS-related services
g) IIS dynamic content
h) SQL services
i) SQL queries
j) ODBC connection

I came to the conclusion that, in order to satisfy the above items, as well as item a from section 3.2.2, the lab had to be comprised of a total of four machines, including a client machine from which to perform tests on the servers (the exact configuration of the lab machines is detailed later). In addition, to satisfy item d from 3.2.2, the entire lab would need to be accessible from the desks of both engineers. However, to keep the lab pure, and to protect the company network (especially since we will be running a DHCP server in the lab), the lab servers could not be logically connected to the company network. Therefore, the infrastructure had to be laid out as in figure 1, on the next page:

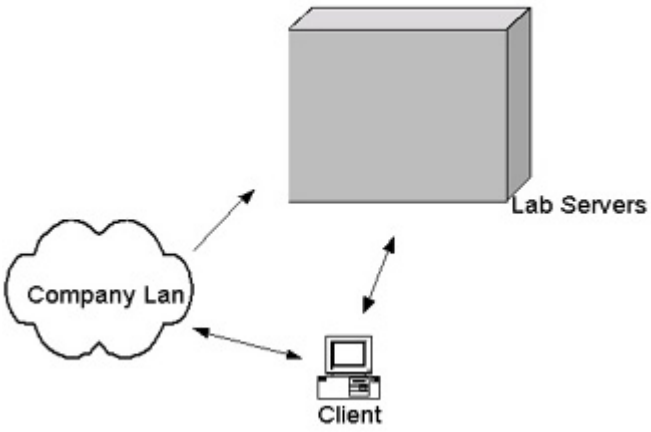


figure 1

Note that in figure 1, the gray box represents three separate lab servers, one running IIS, one running SQL, and the third as a domain controller also running core services WINS, DNS and DHCP.

My first idea was of a VLAN[2] segment between the company network and the lab. The IT engineers would then connect to the client machine from their desktops via VNC[3] and run the tests against the lab servers across the VLAN. This would meet the aforementioned requirements, while providing the additional benefit of keeping the lab infrastructure secure from accidental unauthorized access. A physical VLAN implementation would have looked like figure 2, below:

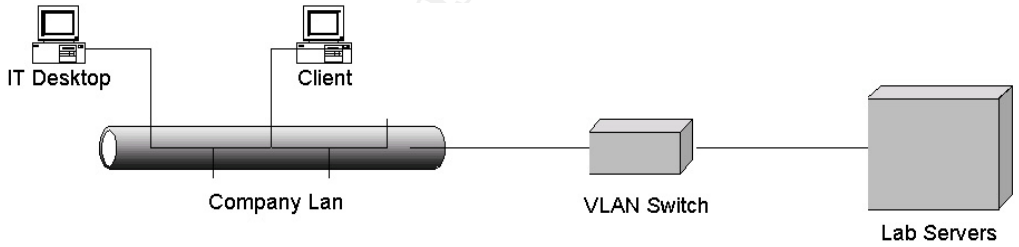


figure 2

This idea had a few weaknesses, however. The first was logistics. Three separate lab servers would require a significant amount of space and, because the machines would have to be backed up, that space would have to be in a main data center managed by IT. Space in a main data center is expensive, and we were trying to keep costs down as much as possible. The second problem was resources. To implement the lab in this way would have required the involvement of several different IT groups, and a fair amount of work. The VLAN would have to be configured by the network group, the data center space would have to be provided by the data center group, and that would have required the production control group to create managed device records for each machine (a

[2] explanation of VLANs can be found at: http://net21.ucdavis.edu/newvlan.htm#what
[3] http://www.uk.research.att.com/vnc/

requirement for machines in the main data centers). The third issue was the significant cost of buying three adequate server machines. Cost is an especially important consideration in a virtual team environment, as there is no cost center associated with virtual teams. The last, and most troublesome, issue was that the test servers would be constantly having patches applied, at least some of which would probably not test out properly. That meant having to back them out. That, in turn, meant having to do occasional restores. This work would have to be done by the data center group, meaning on-going involvement, and an on-going commitment of resources from another IT group.

After considering, and discarding, several modifications to the VLAN idea, I remembered a paper I had recently read on Lance Spitzner's Honeynet Project website regarding the use of VMware to create "virtual" labs[4]. After designing a few different lab layouts, I settled on the one depicted below, in figure 3:

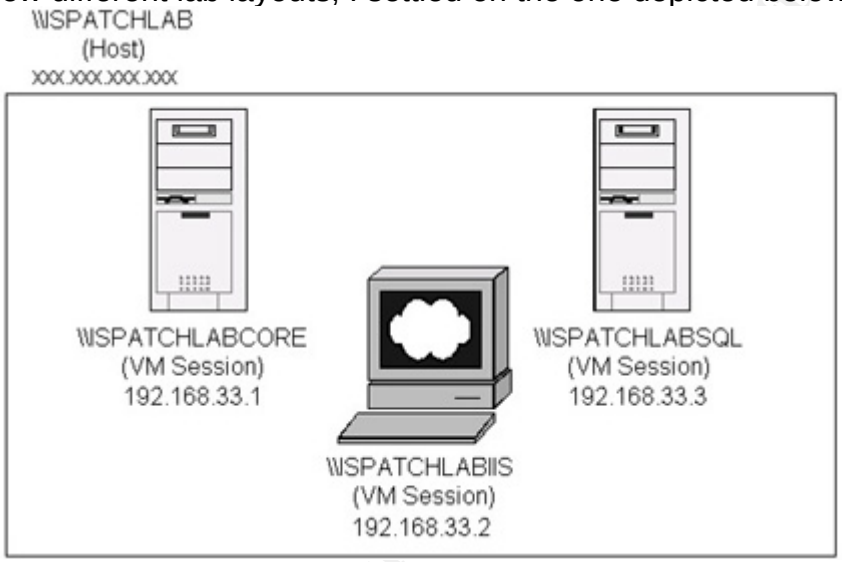


figure 3

Then, by using two NICs in both the client machine and the host server machine, the lab could be logically separated from the company network, while retaining the ability for the IT engineers to use it from their desks (again via VNC), and for the host machine to retrieve the patches to be tested from the Internet. This physical layout is shown on the following page, in figure 4:

[4] http://project.honeynet.org/papers/VMware/

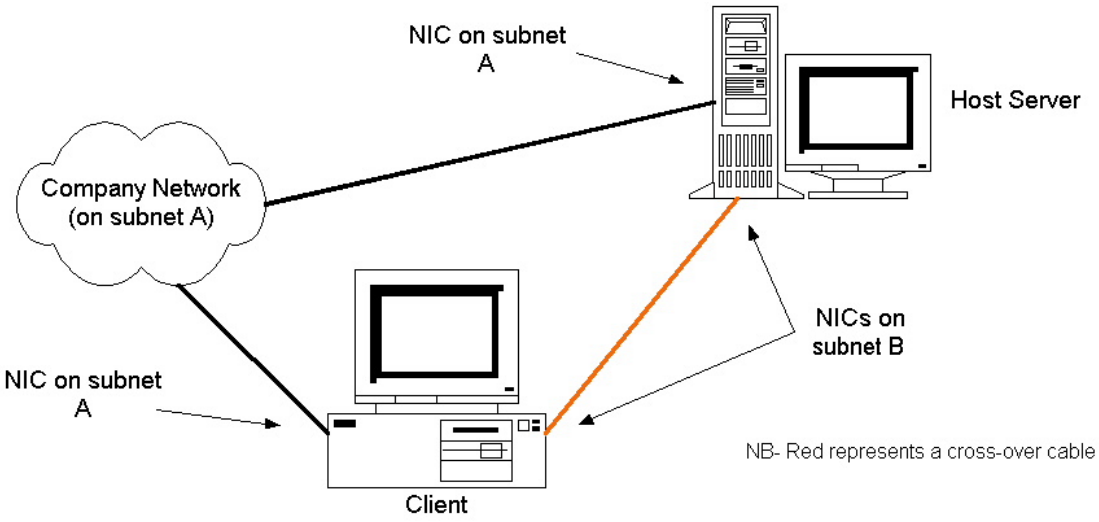


figure 4

Note that, in figure 4, only the host (physical) server will have access to the NIC on subnet A. All the virtual machines created with VMware sit on subnet B and are bridged through the server NIC on subnet B (refer to figure 3).

The VMware-based addressed the shortcomings of the VLAN-based proposal, as listed below:

a) Less space will be used by one physical server than by three
b) A VLAN is no longer required, eliminating the need for involvement of the network team.
c) Since virtual machines are saved as a single file on the host machine, the files can simply be copied off regularly by a scheduled task (to the client machine, in our implementation), as opposed to being backed up as individual systems. For its part, the host server can be occasionally "cloned"[5]. This eliminates the need for the lab to be placed in the main data center, in turn eliminating the logistical problems noted earlier.
d) Although the software cost remains the same, the hardware costs were drastically reduced, theoretically by two-thirds.
e) VMware has a brilliant feature, called a redo log, which allows for changes to be made to the software on the virtual machines *without* being permanently committed. This meant any unwanted changes could be backed out completely and harmlessly after installation, but before being permanently committed to the virtual machine's operating file. This eliminates the need for on-the-spot restores, thus eliminating the need to impose upon the data center group to do them.

Once my latest proposal was approved by the other members of the sub-team, I got to work at procuring hardware and software, and building the lab. Using the

[5] with RAID 1 and 5+1 on Compaq servers (our host is a DL 360), one of the drives can be hot-swapped, and will be quickly rebuilt from parity, allowing the original drive to be kept as a "clone" and stored separately from the server

hardware guidelines from Baker Hart's excellent paper on virtual labs[6], I was able to source a suitable spare Compaq DL 360 server and disk drives from my own IT group (the one I work for when not contributing to the VPT), and one of the other sub-team members was able to source a desktop machine with a standard company image on it to be used as the client. According to Hart's recommendations, we needed 1 GB of RAM, which was the only hardware that had to be bought new, at an approximate cost of $400. It was paid for by the manager who created and co-ordinates the VPT. This ended up being the only invoiced cost of the entire project, as all the software licenses were pre-purchased, including the VMware license, which the same manager was able to obtain from another IT group that had a spare. Once the virtual machines were built, the applications were installed and configured to facilitate the testing, as follows:

Virtual machine 1- \\ISPATCHLABCORE
-Domain Controller for the ISPATCHLAB domain, which contains three machine accounts, one for each of the two other virtual machines, and one for the client.
-DHCP Server service is running a scope of 10 IP addresses, and also provides the DNS and WINS addresses (its own) to DHCP clients.
-DNS service is running with one DNS record in place, for the fictitious machine \\BOGUSRECORD, whose IP is 1.1.1.1
-WINS service is running.

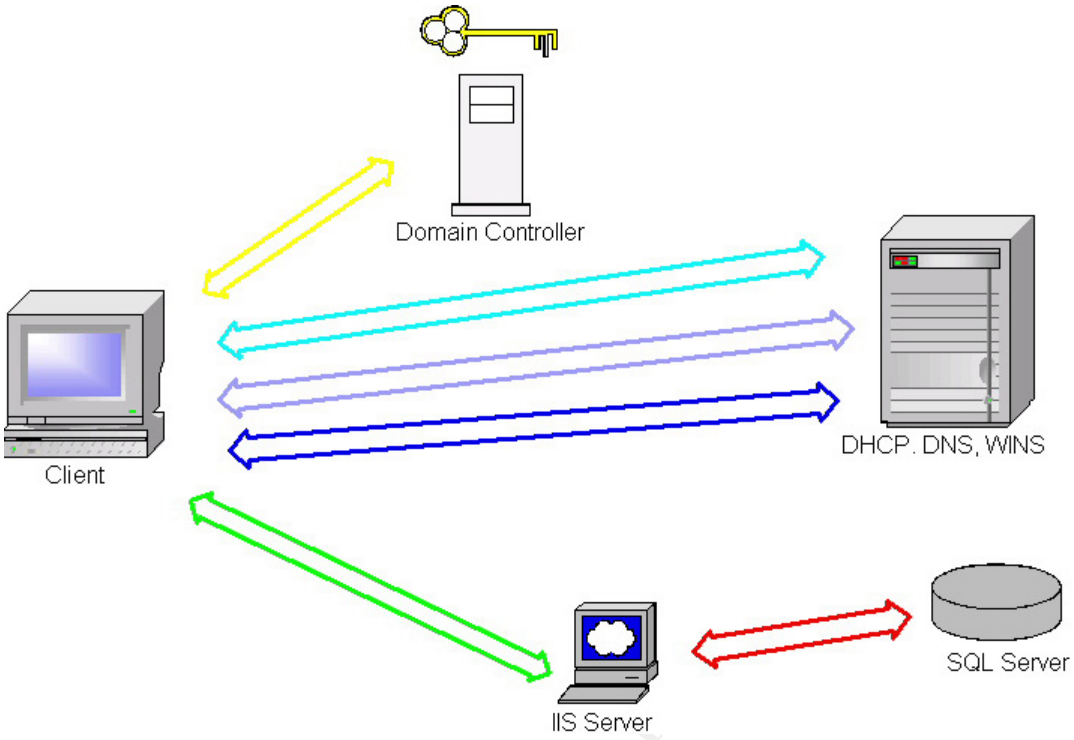
Virtual machine 2- \\ISPATCHLABIIS
-IIS 5 server, serving one .asp page with an input field for querying the SQL database on \\ISPATCHLABSQL
-ODBC connector is installed to query the database on \\ISPATCHLABSQL, using application-level authentication.

Virtual machine 3- \\ISPATCHLABSQL
-SQL 2000 installed running one database, with one table, containing three six-digit numeric strings.

3.2.4 Defining the Testing Script

In order to test for all ten items listed at the beginning of section 3.2.3, a test script was created. The following page is the definitive test script, which is reprinted directly from the documentation I created for the IT engineers doing the testing.

[6] http://www.giac.org/practical/Edwin_Hart_GSEC.doc



1) **Boot the client machine and log on.** A successful logon verifies that a DHCP address has been obtained (figure 3, light blue arrow) and that authentication is working properly (figure3, yellow arrow)
2) **From a command window, enter: *ping bogusrecord.ispatchlab.*** The result should be a resolution to IP 1.1.1.1. This verifies that DNS is working (figure 3, purple arrow)
3) **Open IIS and go to: *http://ispatchlabiis.*** The result should be a page with a query window (titled: Test Page) asking for input. This verifies that WINS resolution is working (figure 3, dark blue arrow) and that IIS is working (figure 3, green arrow)
4) **Enter any single digit *x* in the query window.** The result should be a returned digit indicating the number of digit *x* found in the SQL database. This verifies that SQL is functioning and that the ODBC connection on \\ISPATCHLABIIS is functioning (figure 3, red arrow)

If the Testing Process is Completed Successfully

Note the release date of the patch in question. When one week has elapsed, publish the successful results to the patch database, and commit the changes to disk by powering off the VM Ware machines and choosing “Commit the changes in the redo log to the disk”. The patch can be considered approved when the result is published to the database.

If the Testing Process is Not Completed Successfully

When the testing process cannot be properly completed, or the server(s) exhibit(s) any unusual behavior, detail the issue and note it in the patch database with the result for that patch. Shutdown the VM Ware machines, then do a “power off” and choose “Discard the changes in the redo log” when prompted. The patch can be considered to be failed (not approved) at this time, but should be tested again if another iteration of the patch is released.

3.2.5 Communicating the Results

Once the lab was up and running properly, we needed to publish the results to make them available to the user community. My plan was to provide a web-based search page that could be used to query the results database that was being populated by the testers. The idea was that the page could hang off of the IT portion of the company's Intranet site. One of the members of my sub-team built a simple, but effective, query page which would allow the user to query the specific q article or hotfix number, and would return the following information:

a) q article or hotfix number
b) whether or not it had been tested
c) whether or not it had passed
d) any comments included by the testers (mostly for describing why a patch didn't pass)

The page also contains links to the testing script and a description of the physical and logical layout of the lab itself.

After the page was posted, we had to get the word out to Engineering that the service was available. Another sub-team of the VPT had spent time creating a fairly comprehensive mailing list of known departmental server administrators (about 120 people), and we leveraged this mailing list to advertise the service.

4.0 After Snapshot

Although we came away from the days of Code Red, NIMDA, and Slammer much better off than many of the organizations I have read about, a casual calculation of the cost of our downtime[7] still comes out amazingly high at almost $400k USD for the three combined incidents. At the time of this writing, the initiatives of the latest VPT (including the patch lab infrastructure) have been in place for less than a month. Recent network scans are showing that roughly 65% of the identified departmental servers are patched reasonably up-to-date as opposed to less than 10% at the beginning of the VPT's term. Looked at from another angle, this equates to an almost 10-fold increase in the number of machines in the target demographic that are now protected. As mentioned at the outset of this paper, that demographic was largely responsible for the spread and resultant downtime of Code Red and NIMDA. And, while log results from the Slammer attack aren't yet available to me, I expect similar findings. Although it would be impossible to gauge exactly what effect such a major reduction in unpatched server machines would have on the total dollar figure, it is certain that it would be significant.

While this remarkable gain is a credit to the efforts of the entire Virus Protection Team, I believe that the successful implementation of the patch lab infrastructure

[7] http://www.zdnet.com.au/itmanager/management/story/0,2000029576,20264775,00.htm

was responsible for a considerable piece of this result. The missing key elements, as discussed in section 3.1 have been provided: Insight into the testing process is available, as the testing process is explicitly posted and linked from the patch results query tool. A significant level of responsibility is implied in that the patches are now officially approved by IT. If a patch that we have approved causes a problem, Engineering now knows that we are responsible to help fix it. And with the internal IT department doing the testing, the level of trust in the work will be increased dramatically, especially considering that the users can now see exactly what tests we do, and how we do them.

As well as fulfilling the original goals I identified, and having the potential to produce cost savings by reducing downtime, the patch lab infrastructure has provided several side benefits. The IT department is now able to benefit from a simpler, more comprehensive testing process, better facilities, and a database of test results, as opposed to the somewhat haphazard way testing was performed in the past. This not only increases information assurance for the entire production server infrastructure, but also gives us greater credibility, and a head start on any process auditing initiatives that may materialize. Also, because the testing takes less time and can be shared between the two engineers, frees up those engineers to do other things. Further, this benefit came almost "free" as it was not in the original scope of the project. We also have a higher level of visibility with Engineering, due to a service that we can point to as being provided as a direct result of their needs. Lastly, we have an excellent base upon which we can build this service out to include other OS and application platforms, or even new hardware, when needed.

The immediate increase in patching compliance and the informal feedback I have received tells me that by fulfilling the projects goals, we have solved the original problem. The Engineering staff, or at least a large portion of them, now have that "measure of confidence" required to make them comfortable with patching their systems. There is a phrase which can be seen so often in information security-related magazines and on websites that it is in danger of becoming cliché: "Information Security is not a technological problem, it is a people problem". This project is a perfect example of the truth of that statement. Neither the problem, nor its solution was a matter of technology. Certainly, there were technical pieces; VMware, databases, web pages, etc. But the real solutions, the pieces most visible to the Engineering staff, were process, communication, and responsibility.

5.0 Bibliography

1. VMware Workstation User's Manual
   http://vmware-svca.www.conxion.com/software/ws32_manual.pdf

2. Building a Security Lab with Virtual Machines - Baker Hart
   http://www.giac.org/practical/Edwin_Hart_GSEC.doc

3. Calculating the true cost of downtime – Mike Sisco, April 2002
   http://www.zdnet.com.au/itmanager/management/story/0,2000029576,20264775,00.htm

4. Know Your Enemy: Learning with Vmware – January 2003
   http://project.honeynet.org/papers/vmware/

5. VLAN Information – University of California, Davis
   http://net21.ucdavis.edu/newvlan.htm